Package ‘ggpointdensity’

October 13, 2022

Type Package
Title A Cross Between a 2D Density Plot and a Scatter Plot
Version 0.1.0
Description A cross between a 2D density plot and a scatter plot, implemented as a ‘ggplot2’ geom. Points in the scatter plot are colored by the number of neighboring points. This is useful to visualize the 2D-distribution of points in case of overplotting.

URL https://github.com/LKremer/ggpointdensity
BugReports https://github.com/LKremer/ggpointdensity/issues
License GPL-3 | file LICENSE
Encoding UTF-8
LazyData true
Depends R (>= 3.2)
Imports ggplot2
Suggests viridis, dplyr
NeedsCompilation yes
Author Lukas P. M. Kremer [aut, cre] (<https://orcid.org/0000-0003-3170-6295>), Simon Anders [ctb] (<https://orcid.org/0000-0003-4868-1805>)
Maintainer Lukas P. M. Kremer <L-Kremer@web.de>
Repository CRAN
Date/Publication 2019-08-28 14:30:02 UTC

R topics documented:

  geom_pointdensity .............................................. 2

Index 5
**Description**

The pointdensity geom is used to create scatterplots where each point is colored by the number of neighboring points. This is useful to visualize the 2D-distribution of points in case of overplotting.

**Usage**

```r
geom_pointdensity(mapping = NULL, data = NULL, stat = "pointdensity", position = "identity", ..., na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

**Arguments**

- `mapping`: Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- `data`: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).

- `stat`: The statistical transformation to use on the data for this layer, as a string.

- `position`: Position adjustment, either as a string, or the result of a call to a position adjustment function.

- `...`: Other arguments passed on to `layer()`. This includes `adj`, a multiplicative bandwidth adjustment used to adjust the distance threshold to consider two points as neighbors, i.e. the radius around points in which neighbors are counted. For example, `adj = 0.5` means use half of the default. Other arguments may be aesthetics, used to set an aesthetic to a fixed value, like `shape = 17` or `size = 3`. They may also be parameters to the paired geom/stat.

- `na.rm`: If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

- `show.legend`: Logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

- `inherit.aes`: If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.
geom_pointdensity

Author(s)
Lukas P.M. Kremer

References
https://GitHub.com/LKremer/ggpointdensity

Examples

```r
library(ggplot2)
library(dplyr)
library(ggpointdensity)

# generate some toy data
dat <- bind_rows(
  tibble(x = rnorm(7000, sd = 1),
         y = rnorm(7000, sd = 10),
         group = "foo"),
  tibble(x = rnorm(3000, mean = 1, sd = .5),
         y = rnorm(3000, mean = 7, sd = 5),
         group = "bar"))

# plot it with geom_pointdensity()
ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity()

# adjust the smoothing bandwidth,
# i.e. the radius around the points
# in which neighbors are counted
ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity(adjust = .1)

 ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity(adjust = 4)

# I recommend the viridis package
# for a more useful color scale
library(viridis)

 ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis()

# Of course you can combine the geom with standard
# ggplot2 features such as facets...

 ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis() +
  facet_wrap(~ group)

# ... or point shape and size:

 dat_subset <- sample_frac(dat, .1) # smaller data set

 ggplot(data = dat_subset, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis() +
  facet_wrap(~ group)
```
geom_pointdensity(size = 3, shape = 17) +
scale_color_viridis()

# Zooming into the axis works as well, keep in mind
# that xlim() and ylim() change the density since they
# remove data points.
# It may be better to use `coord_cartesian()` instead.
ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis() +
  xlim(-1, 3) + ylim(-5, 15)

```
ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis() +
  coord_cartesian(xlim = c(-1, 3), ylim = c(-5, 15))
```
Index

aes(), 2
aes_(), 2

borders(), 2

fortify(), 2

geom_pointdensity, 2
ggplot(), 2

layer(), 2

stat_pointdensity(geom_pointdensity), 2
StatPointdensity(geom_pointdensity), 2